SWE 265P
Reverse Engineering and Modeling

Lecture 3

Duplication of course material for any purpose without the explicit written permission of the professor is prohibited.
“Then, I [...] and draw things out. Important classes, what functions all called to perform different features. Usually on a piece of paper.” – Kristina Nasr [Software engineer, Google]
Today

• Last week’s material

• Mental models

• Externalizing mental models

• UML class diagrams

• In-class practice

• Alegria Baquero (Zocdoc)
Last week’s material

• Information foraging
  – top-down, bottom-up, systematic, opportunistic comprehension
  – goal-driven
  – familiarity

• JPacMan
  – “understanding”
  – changes!

• Homework

• Any questions?
Mental model

- An explanation of someone's thought process about how something works in the real world
- A representation of the surrounding world, the relationships between its various parts and a person's intuitive perception about his or her own acts and their consequences
- Can help shape behavior and set an approach to solving problems and doing tasks
Properties of mental models

- Individual
- Uncertain
- Selective
- Flexible
- Dependent
Mental model (software, external)

• Mental models are an artefact of belief
  – users will plan and predict their future actions with a system based on their mental models [as constructed from the visible interface of the software, any documentation of what it does, and other means]

• Designers ideally should anticipate users’ mental models so that their product communicates its function through its form
Mental models are an artefact of belief
- developers will plan and predict their future actions with a system based on their mental models [as constructed from the source code, any documentation of how it works, and other means]

Developers ideally should anticipate other developers’ mental models so that their source code communicates its function through its form
Properties of mental models

- Individual
- Uncertain
- Selective
- Flexible
- Dependent
Limitations of mental models

- Mental models are limited in capacity
- Mental models are prone to forgetting aspects
- Mental models cannot be accessed by others
Externalizing mental models

Reading Code
Externalizing mental models

```xml
<?xml version="1.0" encoding="UTF-8"?>
<channel version="3.0.0">
  <nextMetaDataId version="3.0.0"/>
  <responseConnectorProperties version="3.0.0">
    <responseVariable>None</responseVariable>
    <defaultQueueOnResponses>None</defaultQueueOnResponses>
    <string>Auto-generate (Before processing)</string>
  </responseConnectorProperties>
</channel>
```
Externalizing mental models
What to externalize?
Important frequent question #1

Where in the code is this feature implemented?
<table>
<thead>
<tr>
<th>Folder</th>
<th>File</th>
<th>Method</th>
<th>Relevant?</th>
<th>Relevant how?</th>
<th>Confidence</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Template (part 2: where do we still need to go?)

<table>
<thead>
<tr>
<th>Folder</th>
<th>File</th>
<th>Method</th>
<th>Why?</th>
<th>Priority</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Let’s practice: JPacMan3

• You should still have a clone of JPacMan3, but if not – https://github.com/SWE-265P/jpacman3

• Open the project
J PacMan question #1 (where is this feature)

• Scoring
Let’s practice: J PAC MAN3

- In IntelliJ, install and activate the plug-in simpleUMLce
- Restart IntelliJ
- Open the J PAC MAN3 project
- Go to the project view, right click on nl.tudelft.jpacman, select Add to simpleUML diagram, choose New Diagram, choose a name, and after hitting ok make sure to confirm recursively
JPacMan question #2 (where is this feature)

• Collisions
Homework (team)

• With your team, prepare and print a UML class diagram for your chosen system
  – bring this diagram to class next week

• Decide upon two different features and highlight in the UML class diagram where those two features are implemented
  – use the templates

• Prepare a one-to-two page writeup of how you found where those features are implemented
  – be specific
  – attach your templates
Homework (team)

• Due date: start of class next week

• Submit via a GitHub pull request that create a homework_1 folder in your team’s folder, two files:
  – hw1_<team_name>.pdf
  – hw1.uml_<team_name>.png

• Bring a printed copy of your UML diagram

• Start early
Homework (continued)

• Tutorial: reading code
  – https://www.youtube.com/watch?v=cPVu9AJ8gGw&t=523s

• How to read code
  – https://www.youtube.com/watch?v=-KgU5sxGtuM

• Strategies for working with legacy code
  – https://www.youtube.com/watch?v=UH4dSpPieDE
Homework (continued)

• Make sure to regularly update your personal diary, including an entry for today’s lecture
Optional advanced material

• Download the codecrumbs tool and experiment with how you may be able to use it in externalizing your mental model

• Experiment with different UML tools
  – Star UML
  – UML designer
  – Visual Paradigm
  – ...

SWE 265P – Reverse Engineering and Modeling
And now...

• ...welcome Alegria!